NASA/TM—2013–216602

Dual Level Parallel Computations for Large-Scale High-Fidelity Database to Design Aerospace Vehicles

Guru P. Guruswamy
Ames Research Center, Moffett Field, California

September 2013

NASA/TM—2013–216602

Dual Level Parallel Computations for Large-Scale High-Fidelity Database to Design Aerospace Vehicles

Guru P. Guruswamy
Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

September 2013

Summary

A dual-level parallel procedure is presented for computing large databases to support aerospace vehicle design. This procedure has been developed as a single Unix script within the Parallel Batch Submission environment utilizing MPIexec and runs MPI based analysis software. It has been developed to provide a process for aerospace designers to generate data for large numbers of cases with the highest possible fidelity and reasonable wall clock time. A single job submission environment has been created to avoid keeping track of multiple jobs and the associated system administration overhead. The process has been demonstrated for computing large databases for the design of typical aerospace configurations, a launch vehicle and a rotorcraft.

Introduction

Design of aerospace vehicle configurations involves a large number of computations with varying parameters. For example, designing launch vehicles to fly safely through the transonic regime requires thousands of computations at different Mach numbers, angles of attack, frequencies, and structural modes [1]. Safe design of rotorcraft, typically dominated by flow complexities and non-linear structural behaviors, requires a large number of computations with different flight scenarios [2, 3].

Currently, low-fidelity models are in use to cope with the requirement of rapid turn-around time needed in the design process [4]. For launch vehicles, low-fidelity linear-aerodynamic theory, such as the vortex lattice method for subsonic flows and Mach box theory for supersonic flows are used. In order to accurately model moving shock-waves and flow separations, high fidelity equations, such as those based on the Navier-Stokes equations for aerodynamics [5] and the non-linear structural dynamics equations for structures [6], are needed. Numerical solution of these high fidelity equations is computationally expensive. For example, given the same structural complexity, the ratio of computational time required for Navier-Stokes-based computations to linear aerodynamics computations for a flexible rectangular wing is about 2000 to 1 [7]. This ratio can significantly increase with increasing geometric complexity. Similar massive computations are required in designing launch vehicle structures [8].

In general, engineering design requires solutions for a large number of computationally intensive cases. Typically, the design process is a time-critical series of events, and timely generation of data plays a crucial factor [4]. Therefore, fast computations with reasonable turn around time are essential for the design process.

Cluster computers [9] have facilitated the fast turn around times needed by design engineers. Most high fidelity single-case computations [10] employ message passing interface (MPI) [11]. However, in order to make high-fidelity-based computations practical for design, a procedure that can generate data for a large number of cases in the least amount of time is needed. Some of the earlier efforts to run multiple cases such as AeroDB [12] and GNUparallel [13] were focused on submitting the multiple jobs (one for each case) on multiple computers. The number of cases was limited by the total number of batch jobs allowed on a particular computer. These tools had the ability to monitor jobs remotely, but the input parameters were limited to flow quantities such as Mach number and angle of attack [12]. In addition they depended on net-work performance for fast turn-around time.

Recently, a procedure based on MPIexec [14] which ran multiple cases in a single job on a single image cluster was presented in Ref. [15]. In addition to flow parameters the procedure in Ref. 15 allowed variations of geometry-dependent parameters such as structural mode shapes. MPIExec was used to initialize a parallel job within a Parallel Batch Submission (PBS) environment [16]. MPIexec uses the task manager library of PBS to spawn copies of the executable on the nodes in a PBS allocation. In Ref .15 the focus was to run maximum number cases simultaneously in a single PBS job environment. The number of cases are independent of the maximum number of jobs allowed on the system. Reference 15 demonstrated generation of a 1000-case design database within 135 minutes of wall clock time for a launch vehicle by using the Navier-Stokes equations. Cases considered involved variations in Mach number, angle of attack and structural mode shape. This procedure does not incur system overhead associated with the starting and closing multiple jobs like the procedures presented in Refs. 12 and 13.

Reference 15 did not use MPI. In this paper, the Ref. 15 approach is further extended to accommodate increased complexity of the individual cases using MPI-based analysis software. The process is demonstrated by generating data for two aerospace configurations, a launch vehicle and a rotorcraft. Parametric studies that not only vary flow parameters but also include variations in structural mode shapes and frequencies are performed. Running multiple cases in a single job environment will facilitate communications directly between two cases when needed.

Approach

An important step in the engineering design is an iterative process that involves computations of a repetitive nature for varying input parameters. These computations take the same amount of time for each case. For example, to design a launch vehicle; a large database is required for a given range of Mach numbers, angles of attack, structural frequencies and modes [17]. It is assumed that all cases are numerically similar and take the same amount of time.

In general, the database generation process for design can be expressed as a multidimensional matrix assigning indices for each parameter. As an example, for the case of a launch vehicle i, j, and k represent Mach number (M), angle of attack (α), and mode shape(Φ), respectively. The total normal force on a launch vehicle at the ith M, the jth α and the kth mode is represented by $CN_{ijk.}$

In this work a run matrix for design data will be generated using PBS with many simultaneous executions of MPI based analysis software. This involves building logic in the PBS script to fetch the appropriate inputs. Using matrix representation of the database needs multiple loops in the script, which can introduce overhead. As a result, a contiguous representation of data is used. The element e in contiguous representation corresponding to an element in matrix is defined [18] as

$$e = \sum_{r=1}^{n} p_r \left(\prod_{q=r+1}^{n} d_q \right) \tag{1}$$

where n is the number of dimensions, d_q is the size of q^{th} dimension, p_r position in the matrix corresponding to r^{th} dimension. The second large symbol above is an upper-case Greek letter pi. It indicates the product of many factors in the same way that an upper-case sigma indicates the sum of many terms. Pi stands for "product," sigma for "sum." For example in a three dimensional matrix case (n = 3) of size 8 x 10 x 6 ($d_1 = 8$, $d_2 = 10$ $d_3 = 6$), the element at position $<p_1 = 2, p_2 = 4, p_3 = 1>$ results in an index in contiguous representation e = 2x(10x6) + 4x6 + 1x1 = 145. Equation 1 is used in PBS scripts to fetch appropriate data from disk.

In the design of an aerospace vehicle the use of an aerodynamic influence coefficient matrix (AIC) [19] is very common. The AIC matrix [A] represents the changes in the aerodynamic loads due to perturbations of structural deformations. Deformations are represented in the form of modes [20]. A_{ij} represents the modal force on the i^{th} mode due to structural perturbation in the j^{th} mode. This allows the designer to compute the loads needed for any deformation in the design process using

$$\{d\} = [\Phi]\{h\} \tag{2}$$

where {d} represents displacements, [Φ] is the modal matrix and h represents generalized displacements.

The modal force vector can be computed using

$$\{F\} = [A]\{h\}. \tag{3}$$

Now the rest of the challenge lies in computing [A] for various parameters. Based on low-fidelity methods, computing [A] is well automated in codes such as NASTRAN [21]. However the accurate analysis for design of aerospace vehicles needs high fidelity Navier-Stokes flow equations. In this effort the OVERFLOW [22] code is used to solve the Navier Stokes equations.

In order to accomplish fast generation of data using high fidelity equations, a script that uses MPIexec with MPI based parallel codes has been developed. In this script, a certain number of cores are dedicated to each case to facilitate use of MPI. The number of cores assigned to each case is termed as rank count (RC). All cases are run in parallel using the MPIexec utility.

The above procedure has been implemented on the Pleiades super cluster that has Hapertown (HAR), Nehlam (NEH), Westmere (WES) and Sandy Bridge (SAN) nodes [23]. The maximum number of cores permissible and associated memory are given in Table 1. Since both HAR and NEH have 8 cores, only HAR is selected in this study.

Table 1: Nodes on Pleiades Supercluster

Node	**Cores**	**Memory/Node**	**Speed in GHz**
Harpertown	8	8GB	3.00
Nehalem	8	3GB	2.93
Westmere	12	24GB	2.93
Sandy-Bridge	16	32GB	2.6

Present approach is aimed towards generating [A] of Eqn. (3). For this a given configuration is represented by several modes. The effects of modal perturbations are independently computed. By using a linear combination of modal loads, aerodynamic forces for any arbitrary displacement can then be computed using Eq. (2). The following approach is taken to compute the AIC matrix using CFD code:

1. For a given configuration a suitable CFD grid is generated.
2. Using an algebraic approach, deformations due to mode shapes are superimposed on the base grid of the rigid configuration.
3. Inputs for different cases with varying angles of attack (AOA), Mach numbers, frequencies and modes are generated based on user specifications. It is assumed number of iterations specified is adequate for convergence for all cases.
4. Data is spawned to different directories that are contiguously numbered.
5. All cases are computed running each case on a different group of cores using 'Rank' identification.
6. Wait till all jobs are complete. Successful completion of all jobs can be tracked by monitoring the size of the residual files. Once PBS job is successfully competed all residual files will be of the same size. It is assumed that user has selected appropriate parameters such that results converge at the end of the job completion.
7. After all jobs are successfully completed, [A] matrix is assembled.

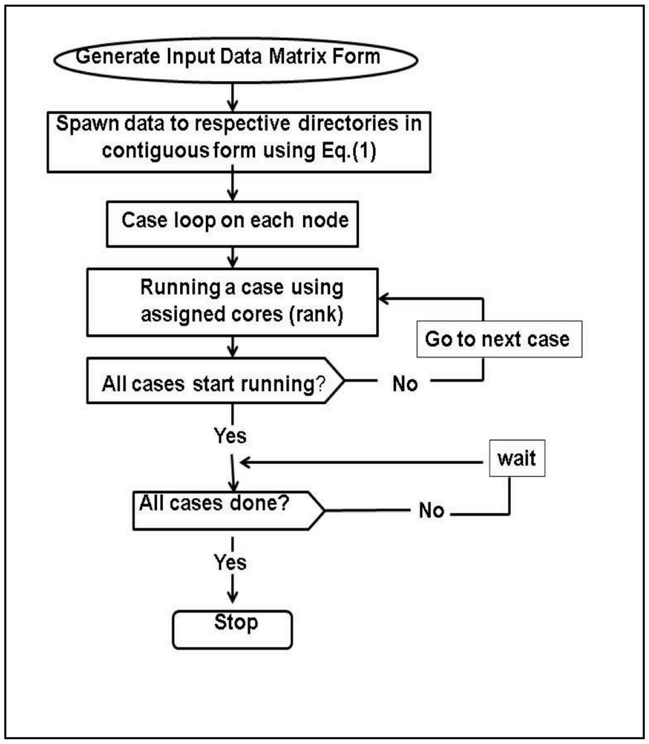
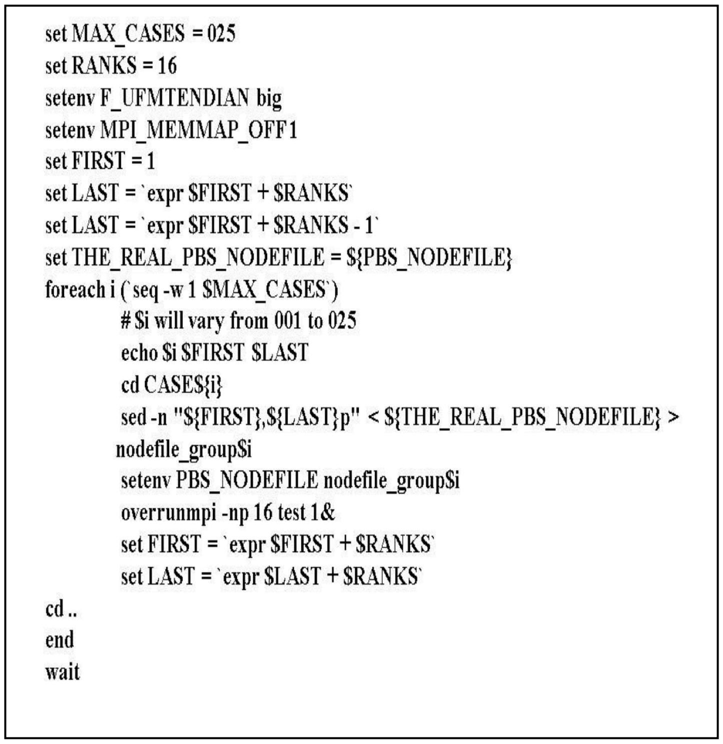

Generate Input Data Matrix Form
Spawn data to respective directories in contiguous form using Eq.(1)
Case loop on each node
Running a case using assigned cores (rank)
Go to next case
All cases start running?
No
Yes
wait
All cases done?
No
Yes
Stop

Fig.1 A flow chart of dual level parallel process RUNDUA



```
set MAX_CASES = 025
set RANKS = 16
setenv F_UFMTENDIAN big
setenv MPI_MEMMAP_OFF1
set FIRST = 1
set LAST = `expr $FIRST + $RANKS`
set LAST = `expr $FIRST + $RANKS - 1`
set THE_REAL_PBS_NODEFILE = ${PBS_NODEFILE}
foreach i (`seq -w 1 $MAX_CASES`)
        # $i will vary from 001 to 025
        echo $i $FIRST $LAST
        cd CASE${i}
        sed -n "${FIRST},${LAST}p" < ${THE_REAL_PBS_NODEFILE} >
       nodefile_group$i
        setenv PBS_NODEFILE nodefile_group$i
        overrunmpi -np 16 test 1&
        set FIRST = `expr $FIRST + $RANKS`
        set LAST = `expr $LAST + $RANKS`
cd ..
end
wait
```

Fig 2 The script for the core of the RUNDUA for 25 case run with RC =16.

A flowchart of the above process called RUNDUA, based on a single Unix script in the PBS environment is shown in Fig. 1. The main script for the RUNDUA is given in Fig. 2. PBS_NODEFILE is the name of the file that contains the list of the nodes assigned to the job. CASE${i} is the directory assigned for ith case and 'overrunmpi' is a special script to that executes the OVERFLOW [22] code.

Demonstrations

The RUNDUA script is demonstrated for National Launch System (NLS) [24] and HART II rotorcraft [25]. Flow computations are made using the overset-grid based OVERFLOW code that solves the Navier-Stokes equations. Previously validated grids are used in this work. Structural flexibilities of configurations are modeled using mode shapes.

Demonstration for Launch Vehicle

A schematic diagram of the NLS configuration [24] is shown in Fig. 3. The grid selected for this configuration has 151, 121 and 50 points in the axial, circumferential and radial directions, respectively. This gird was used in Ref. 15 where the computed results were compared with experiment.

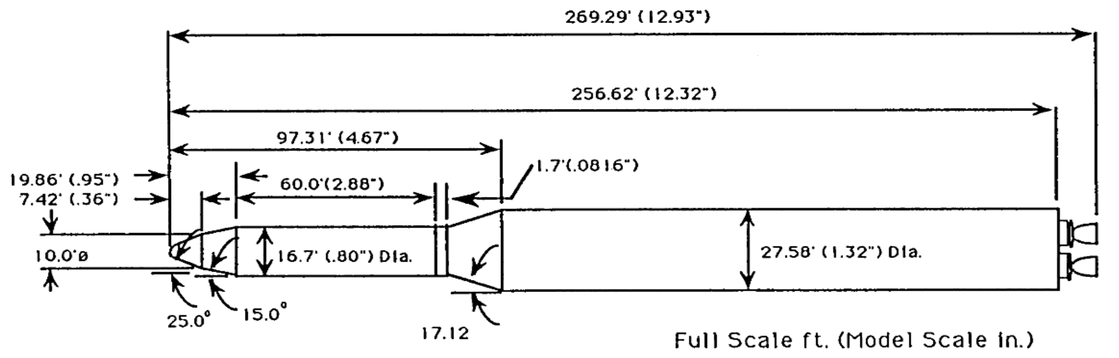


Fig. 3 National Launch System (NLS).

Computations are made for 1,000 cases involving 20 Mach numbers (M_∞) ranging from 0.75 to 1.225, and 10 angles of attack (α) ranging from 0.0 to 4.5 degrees, and 5 modes (1 rigid and 4 flexible modes). Figure 4 shows the distribution of pressure coefficient c_p for the first bending mode at $M_\infty = 0.90$ and $\alpha = 0.0$ deg.

First the performance of the HAR, WES and SAN nodes are studied for a single case. Each case is run for 2000 time steps as required for a converged solution on the deformed configuration [15]. Figure 5 shows the wall clock time in microsecs required per step per grid point. Computations are made using RC of 1, 4 and 8 for a HAR node. Then it is switched to a WES node for RC = 12, followed by a SAN node for RC =16.

On the HAR node, computations speed-up by a factor of about 3 for the increase in RC from 1 to 4 and by a factor of about 2 when RC increases from 4 to 8. Switching to RC = 12 on WES gives another speed-up of 1.5. Finally a factor of 1.25 speed-up is obtained for RC = 16 on the SAN node. The rate of gain in the speed is significantly reduced after RC = 4. This can be attributed to the small grid size used in this specific demonstration case. However solutions using small grids are needed in the preliminary design stage that requires large number of cases.

The focus of this paper is to generate data for maximum number of cases with the highest possible fidelity within a reasonable turn around time. To obtain a one day turn around, a request for a total number of cores of about 4000 or less is feasible for the Pleiades super cluster [15, 22]. From Fig. 5 it is seen that the rate of improvement in speed decreases significantly after RC = 4. Therefore for this case RC = 4 is used for the rest of the computations. Results are demonstrated using the HAR node. Each node will run 2 cases without wasting cores since RC = 4 is a factor of number of cores in the HAR node.

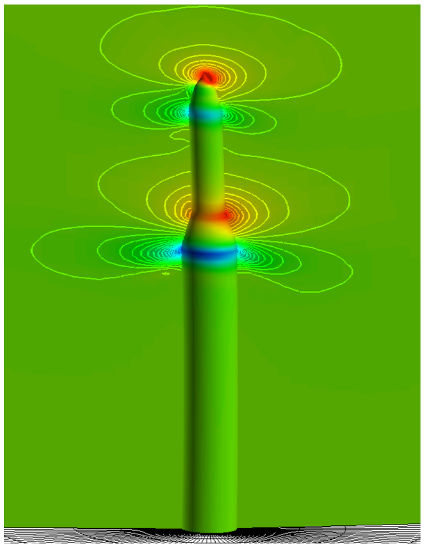


Fig. 4 Cp contours at $M_\infty = 0.90$, $\alpha = 0.0$ deg.

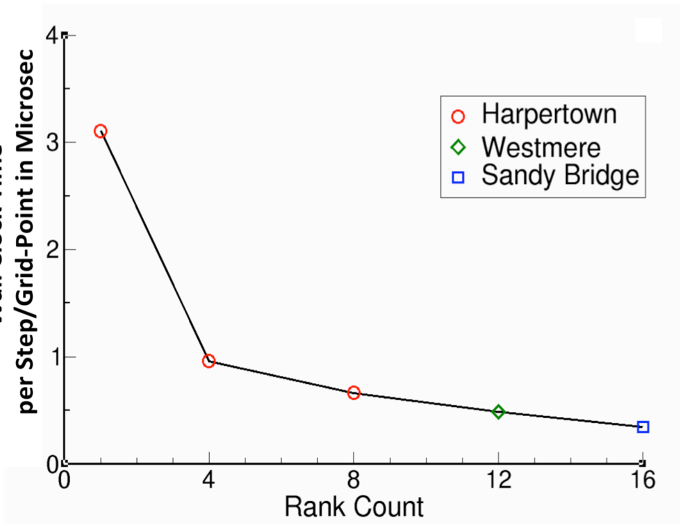


Fig. 5 Performance on different nodes

Changes in M_∞ and α are made in input for OVERFLOW and changes in mode shapes are built into the grid. ‘input_{ij}‘ represents input for ith Mach number and jth angle of attack. ‘grid_k‘ represents the grid for the kth mode. The case number in a contiguous numbering scheme required for RUNDUA is computed using Eq. (1) by setting n = 3 and replacing p_1, p_2 and p_3 by i j and k respectively. For this case the values of d_1, d_2 and d_3 in Eq. (1) are 20, 10 and 5, respectively.

Single job runs are made using RUNDUA in increments of 100 cases up to 1000. On each node a case is run for 2000 steps. Number of steps required for computations is determined based on the convergence of the axial force corresponding to the second bending mode. For example at $M_\infty = 0.90$ and $\alpha = 2.5$ degrees the axial force changed less than 0.1 percent from 1900 to 2000 steps.

A stand alone case using 4 cores required 27 minutes of wall clock time. From run to run, the number of angles of attack are increased from 1 to 10. Figure 6 shows a plot of the percentage increase of wall clock time with respect to the wall clock time required to run a single case. The increase is gradual and reaches 15% for 1000 cases using a wall clock time of 31 minutes. This increase is attributed to internal system time in grouping the cores with the increase in number of cases. Even with a 15% increase in wall clock time, 1000 cases can be completed in a reasonable time. Similar computations without using MPI required a wall clock time of 135 minutes as reported in Ref. 13. The conventional approach of submitting multiple PBS jobs will have over head associated with start and ending jobs that user does not keep track of it.

The results from 1,000 cases are summarized in Fig. 7. Plots of normal (side) force as a function of M_∞ and α are shown for the rigid case and the first 4 flexible modes. All modes have significant influence on the normal forces. The effect is more pronounced near $M_\infty = 1.0$. The rate of change in normal force near transonic Mach numbers is more pronounced for higher modes. Data from such plots are used in designing launch vehicles.

To compare with conventional approaches an attempt was made to run 1000 individual cases similar to that in Ref. 12 as 1000 separate PBS jobs. However, system restrictions on Pleiades [23] did not allow more than 300 PBS jobs.

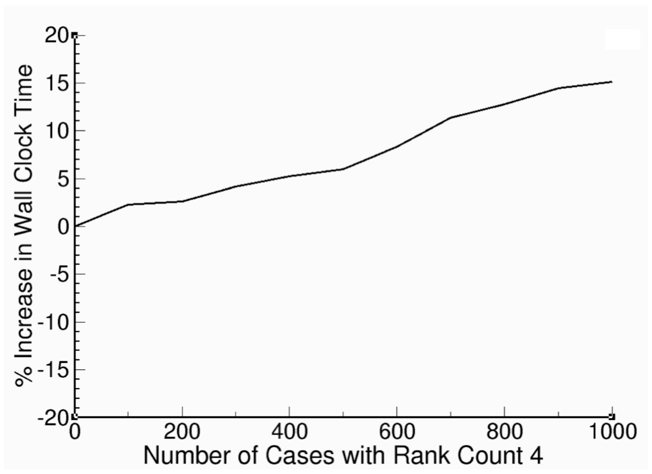


Fig. 6 Percentage increase in wall clock time of PBS jobs compared to a PBS job with single case for NLS configuration using rank count 4 on HAR node.

Rotorcraft (HART-II)

For the next demonstration, computations are performed for the HART-II rotorcraft model [25] shown in Fig. 8. Blades are 2 meters long with 0.121m chord and consist of NACA 23012 airfoil sections. Each blade has a built-in linear twist of -8 deg and a square tip. The detailed structured grid for the OVERFLOW code was generated by Doug Boyd of NASA Langley [5]. Based on that grid, time accurate aeroelastic computations were made with 40 million grid points [3]. The solution required about 30 hrs of wall clock time with 1024 Harpertown cores for 4 revolutions. In this paper, computations are made for the blade-system without the body. Blade grids from the original grid [5] are used.

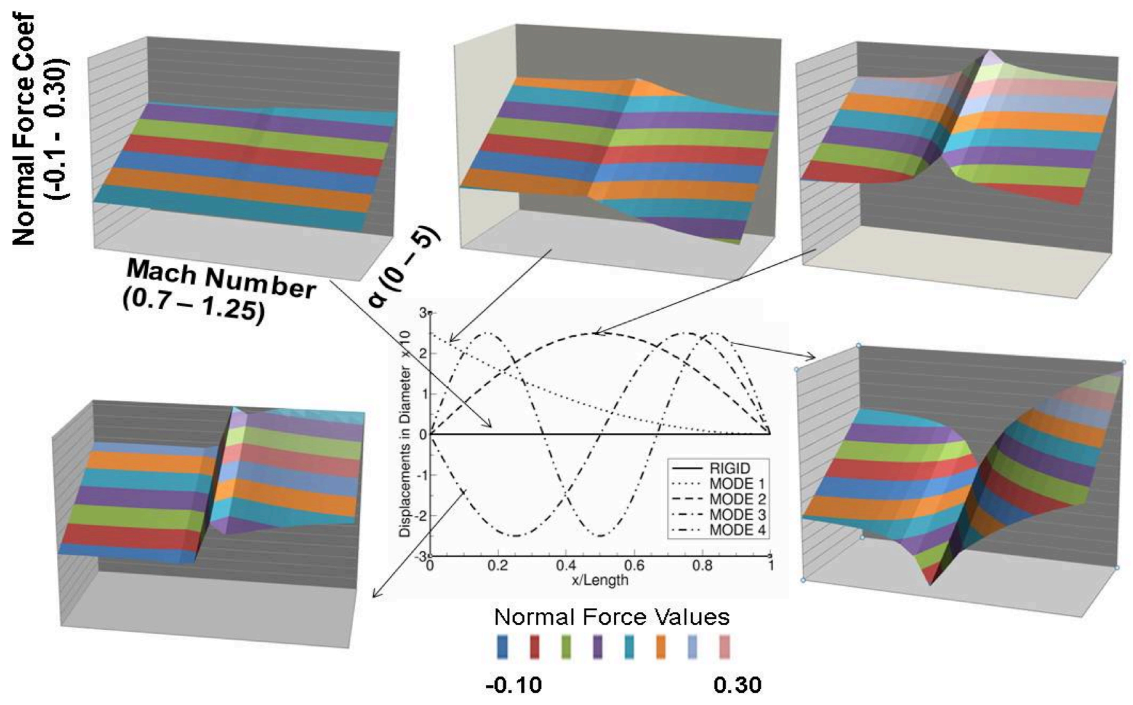


Fig.7 Design database for Launch Vehicle, normal force coefficient versus Mach number M_∞ and angle of attack α.

Unlike in Ref. 3 where the focus was on detailed computations for a single case, in this effort the focus is to generate data for a large number of cases.

In order to understand flight conditions that may lead to dynamic instabilities, it is necessary to compute flutter boundaries of rotorcraft [26]. Computation of flutter boundaries that involves solving Eigen value equations needs a large database involving different rotating speeds, modes, and frequencies. Recently, a procedure to use Navier-Stokes-based aerodynamic data for computing flutter speeds associated with a single isolated rotorcraft blade was presented in Ref. 27. Using MPIexec-based, single-level parallel computations it was demonstrated that a flutter boundary needing data for 10 rotating speeds, 2 modes and 5 frequencies can be computed within about 25 hrs of wall clock time using 100 cores on the Pleiades super cluster. The grid used in that study consisted of 1.8M grid points. In this paper the previously introduced RUNDUA will be demonstrated to compute data for flutter computations for a 4 bladed-system.

Based on the grid from Ref. 5, the total grid size needed for four bladed rotor system is 20 million grid points. In order to test the adequacy of the grid, computations were made for base line conditions [25] (forward speed 200 ft /sec) using the aeroelastic deformations computed in Ref. 3. The solution required 3 revolutions with 3600 time steps per revolution for results to reach a periodic motion. Resulting integrated air loads compared well with the experiment [25] similar to that reported in Ref. 3. Figure 9 shows a plot of surface pressures on the blades. Figure 10 shows a snapshot of vorticity magnitude contours that were generated over 3 revolutions. The development of tip-vortices and blade-wake structure can be seen in Figure 10. This detailed predictive capability of the Navier-Stokes equations used in the present simulation helps to generate accurate data for stability computations including the effects of complex flow phenomena such as blade-vortex interactions, dynamic stall etc.

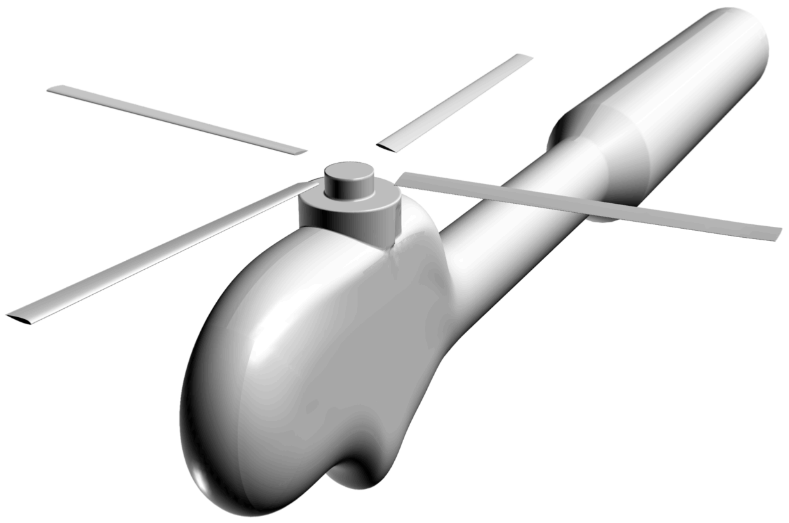
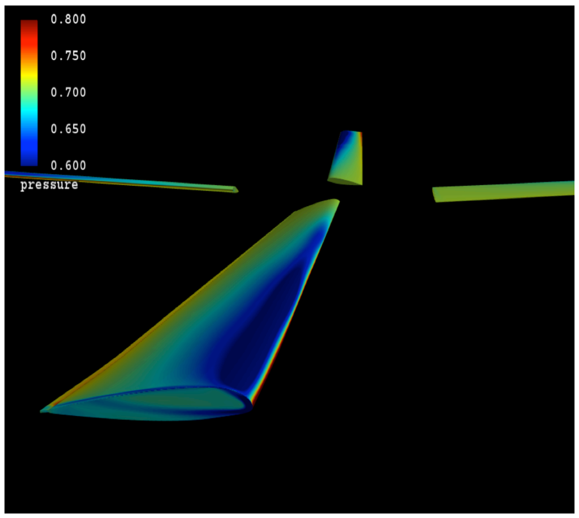


Fig. 8 HART-II rotorcraft.

For computing flutter boundaries several flow computations at various rotating speeds, modes and frequencies are needed with the use of the Navier-Stokes equations. To compute such data, a fast procedure will be demonstrated using RUNDUA.



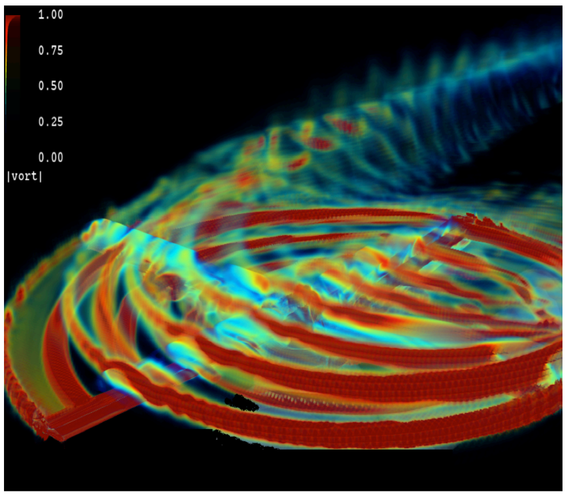


Fig. 10 A plot of vorticity-magnitude for blade-system of HART II.

Since the grid is larger than that for the launch vehicle case, the SAN node that allows a maximum RC of 16 per node is selected. The value of RC required is determined based on computations for a PBS job with single case. Figure 11 shows a plot of wall clock time needed per grid point per step. There is a continuous improvement in speed from RC = 1 to 40. It was decided to use RC = 32 (a multiple 16 corresponding to number of cores on SAN) for further computations to keep the wall clock time around 15 hrs. The RC value is selected as a multiple of 16 to avoid splitting of the node between cases, which adds communication overhead. A single 100-case PBS job is possible assuming a request for 3200 cores is feasible for less than one-day turn-around time.

Input data for 10 rotating speeds ranging from 55 to 95 radians/sec and 2 modes with 5 frequencies are prepared. The changes in rotating speed are made in the input for OVERFLOW. The changes in frequencies for different modes are made through an external input data called the 'motion file' of the OVERFLOW code [27]. Figure 12 shows a plot of tip responses to the 5 torsional frequencies considered.

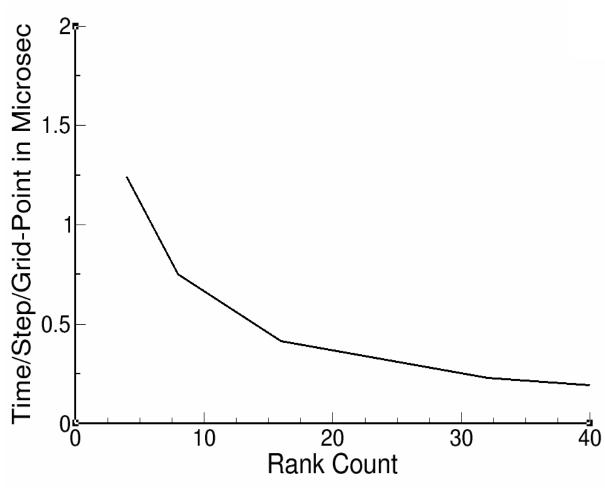


Fig. 11 Effect of rank count on wall clock time for HART-II on SAN nodes.

Unsteady 100-case computations are performed for three rotations with 3600 steps per revolution when responses reached a periodic motion. A wall clock time of 23.15 hrs was required to complete all 100 cases, which is 1% above that required to run a single case.

Fourier transformations of force outputs from OVERFLOW are performed to extract data useful for flutter computations [27]. This process is built into RUNDUA using an Unix script and FORTRAN post processing modules (see Appendix A). Use of Eq. (1) is embedded I the script. Figure 13 shows plots of sectional lift magnitude at 20, 40, 60 and 80 percentage of the blade radius for varying rotational speeds and 5 torsional modes. Similar plots can be extracted for other aerodynamic quantities such as moments. Quantities from these plots can be used as input into the AIC matrix [A] of Eq.(8) in Ref. 27 to compute flutter speed.

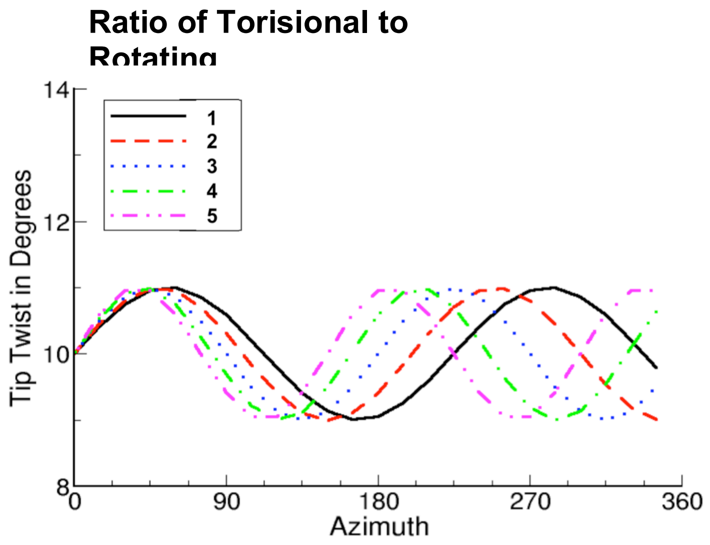


Fig. 12 Tip twist responses of HART II blades associated with 5 frequencies.

In order to compare the turn around time between the present single job environment and the multiple job environment used elsewhere, computations were made for HART-II by submitting one job for each case. The present approach took 5% less turn-around time. However, the turn around time is susceptible to system load and job priorities, which could vary. Therefore, turn-around time can be used as a secondary guideline but not as the prime criteria to judge the efficiency of the process.

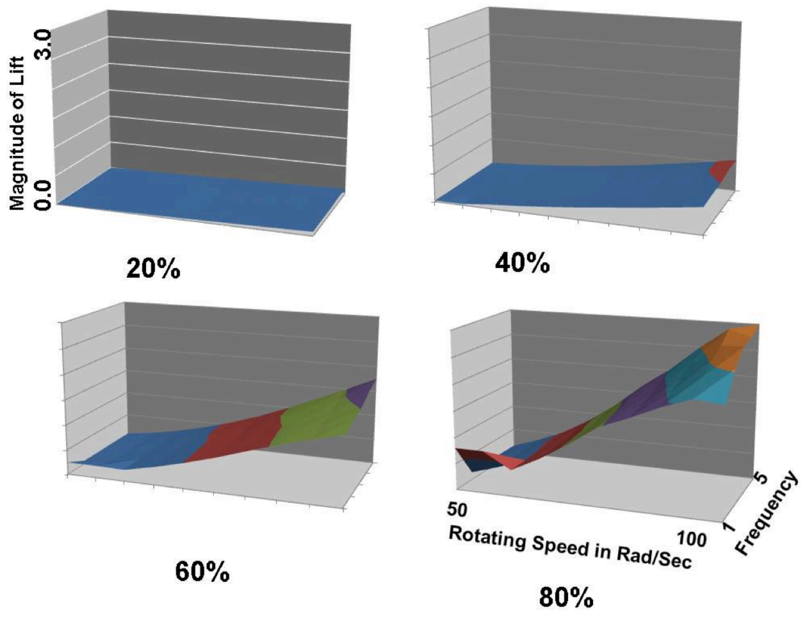


Fig. 13 Effect of rotating speed and torsional frequency on predicted sectional lift.

Conclusions

A dual-level parallel procedure has been developed for computing large databases to support aerospace vehicle design. The procedure has been developed as a single Unix script in the Parallel Batch Submission environment by utilizing MPIexec to run MPI-based analysis software. The procedure provides a single-job submission environment for the user and caters to the needs of aerospace design engineers who need to run a large number of cases using software based on high fidelity methods. Results have been demonstrated for two typical aerospace configurations, a launch vehicle and a rotorcraft. Using a moderate size grid of about 1 million points, it has been demonstrated that 1000 cases can be run within about half an hour of wall clock time to generate preliminary design data for a launch vehicle. For a 4-bladed rotorcraft system using about 20 million grid points, it has been demonstrated that 100 cases can be run within about 15 hrs of wall clock time. For both cases, it was assumed that a request of 4000 cores was feasible to guarantee less than a one-day turn-around. A demonstration case showed that turn around time of this approach was similar to that of submitting many individual jobs but does not involve overhead of system tasks associated with the start and end of multiple jobs. The number of cases that can be run at a time is not limited in the present approach. Unlike previous approaches this single-job environment approach was suitable for extending to three-level parallel computations that include direct MPI communications among different application codes.

Acknowledgements

The author thanks David Barker and Johnny Chang of NASA Advanced Supercomputer Division of Ames Research Center for providing valuable suggestions in developing the script.

References

1) Balesdent, M., Bérend, N., Dépincé, P. and Chriette, A., "A Survey of Multidisciplinary Design Optimization Methods in Launch Vehicle Design," *J. of Structural and Multidisciplinary Optimization,* Vol. 45, Issue 5, May 2012, pp. 619-642.
2) Nash, J. "Performance Optimization of Helicopter Rotor Blades," NASA TM 1040594, April 1991.
3) Guruswamy, G. P., "Fluid/Structure Interaction Modeling of Helicopters Using the Navier-Stokes equations," AIAA 2012-4789, AIAA Modeling and Simulation Technologies Conf., Minneapolis, MN, August 2012.
4) Holleis, P. and Schmidt, A., "MAKEIT: Integrate User Interaction Times in the Design Process of Mobile Applications," Pervasive 2008, LNCS 5013, 2008, pp. 56–74. [http://www.pervasive2008.org]
5) Boyd, D. D., "HART-II Acoustic Predictions using a Coupled CFD/CSD method," American Helicopter Society 65th Annual Forum, May 2009, Grapevine, Texas.
6) Annet M. S., Polanco M. A., "System-Integrated Finite Element Analysis of a Full-Scale Helicopter Crash Test with Deployable Energy Absorbers," American Helicopter Society, 66th Annual Forum, Phoenix, AZ, May 2010.

7) Byun, C., Farhangnia, M. and Guruswamy, G. P., "Aerodynamic Influence Coefficient Computations Using Euler/Navier-Stokes Equations on Parallel Computers," *AIAA J.,* Nov. 1999, Vol. 37, No. 11, pp. 1393-1400.
8) Uhrig, G. and Boury, D., "Large Space Solid Rocket Motors in Europe – Past and Furure Development,"AIAA-98-3980, 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conf & Exhibit, Cleveland, OH, July 1998.
9) Buyya, R. "High Performance Cluster Computing: Architectures and Systems," Vol. 1, ISBN 0-13-013784-7, and Vol. 2, ISBN 0-13-013785-5, Prentice Hall, NJ, USA, 1999.
10) Jespersen, D. C., Pulliam, T. and Buning, P.G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.
11) "Message Passing Interface, MPI," A Message-Passing Interface Standard, University of Tennessee, May 1994.
12) Rogers, S. E., Aftosmis, M. J., Pandya S. A., Pandya, S .A., Chaderjian N. M., Tejnil, E. T and Ahmad, J. U., "Automated CFD Parameter Studies on Distributed Parallel Computers" AIAA Paper 2003-4229, 16th AIAA Computational Fluid Dynamics Conf, Orlando, FL, June 2003.
13) "GNU Operating System" http://www.gnu.org/software/parallel/
14) Fineberg, S., "MPIRUN: A Loader Multidisciplinary and Multizonal MPI," NASA Advanced Supercomputing Division Newsletter, Vol. 2, No. 6, Nov.–Dec. 1994.
15) Guruswamy, G. P.,"Large-Scale Computations for Stability Analysis of Launch Vehicles Using Cluster Computers," *J. of Spacecraft and Rockets*, Vol. 48, No. 4, July-August 2011, pp. 584-588.
16) Stanzione, D., "LSF/PBS Scripting Nuts and Bolts," *Cluster World*, Vol. 2, No. 7, July 2004.
17) Balesdent, M., Bérend, N., Dépincé, P. and Chriette, "A survey of multidisciplinary design optimization methods in launch vehicle design," *Structural and Multidisciplinary Optimization*, Vol. 45, No. 5, 2012, pp. 619-642.
18) Drake, P., "Data Structures and Algorithms in Java," Pearson Prentice Hall Pearson Education, Inc., Upper Saddle River, New Jersey 07458, 2004 (section 12.3, pp. 337-338).
19) Appa, K. and Somashekar, B. R., "Application of Matrix Displacement Method in Study of Panel Flutter," *AIAA J.*, Vol. 7, No. 1, 1969, pp. 672-675.
20) Dugundji, J., "On the Calculation of Natural Modes of Free-Free Structures," *J. of Aero-Space Science*, Vol. 28, No. 2, Feb. 1961, pp. 164-165.
21) "NASTRAN User's Manual," NASA SP-222 (08), June 1986.
22) Nichols, R. H., Tramel R. W. and Buning P. G., "Solver and Turbulence Model Upgrades to OVERFLOW2 for Unsteady and High-Speed Applications," AIAA-2006-2824, AIAA 36th Fluid Dynamics Conference, San Francisco, CA, June 2006.
23) http://www.nas.nasa.gov/hecc/resources/pleiades.html (date accessed Dec 17th, 2012).
24) Springer, A. M. and Pokora, D. C., "Aerodynamic Characteristics of the National Launch System (NLS) 11/2 Stage Launch Vehicle," NASA TP 3488, May 1994.
25) Van der Walla, B. G, Burleyb, C. L., Yuc, Y., Richard, H., Pengele, K. and Beaumierf, P., "The HART II Test–Measurement of Helicopter Rotor Wakes," *Aerospace Science and Technology*, Vol. 8, Issue 4, June 2004, pp. 273-284.

26) Johnston, R. A, “Rotor Stability Prediction and Correlation with Model and Full-Scale Tests,” *J. of the American Helicopter Society*, Vol. 21, No. 2, April 1976, pp. 20-30.
27) Guruswamy, G. P., “Large Scale Aeroelastic Data for Design of Rotating Blades using Navier-Stokes Equations,” AIAA-2012-5715, AIAA 14th Multidisciplinary Conf., Indianapolis, IN, Sept. 2012.

APPENDIX-A – List of RUNDUA script as applied to demonstration cases.

```
#!/bin/csh
#PBS -l select=250:ncpus=16:model=san,walltime=08:00:00
#PBS -W group_list=****
#PBS -q normal
set echo on
module purge
module load comp-intel/2011.7.256
module load mpi-sgi/mpt.2.06r6
cd /nobackupp1/gguruswa/HART2/BLADES/DLEV/EXPORT
rm -r -f CASE*
set MAX_CASES = 099
set n1 = 0
set n2 = 9
set n3 = 0
foreach i (`seq -w 0 $MAX_CASES`)
  mkdir CASE${i}
  cd CASE${i}
  cp ../grid* .
  cp ../mixsur.fmp .
  cp ../xrays* .
  cp ../overrunmpi .
  cp ../overflowmpi .
  cp ../*xml .
  echo "$i $n1 $n3"
  cp ../mode00$n1.txt  motion.txt
  cp ../input$n3 test.1.inp
  cd ..
  if( $n1 == $n2) then
     set n1 = -1
    @ n3 = $n3 + 1
  else
  endif
 @ n1 = $n1 + 1
  end
  wait
  set RANKS = 40
  setenv F_UFMTENDIAN big
  setenv MPI_MEMMAP_OFF 1
  set FIRST = 1
   set LAST = `expr $FIRST + $RANKS - 1`
   set THE_REAL_PBS_NODEFILE = ${PBS_NODEFILE}
  set spd = 0
  set fre = 0
  foreach i (`seq -w 0 $MAX_CASES`)
           # $i will vary from 001 to 025
           echo $i $FIRST $LAST
           cd CASE${i}
           sed -n "${FIRST},${LAST}p" < ${THE_REAL_PBS_NODEFILE} >
nodefile_group$i
           setenv PBS_NODEFILE nodefile_group$i
           overrunmpi -np 40 test 1&
       set FIRST = `expr $FIRST + $RANKS`
       set LAST = `expr $LAST + $RANKS`
cd ..
end
wait
foreach i (`seq -w 0 $MAX_CASES`)
       cd CASE${i}
       cp rotor_1.onerev.txt ../RESULTS1/force$spd$fre
       @ fre = $fre + 1
       if ( $fre == $n2)  then
       set fre = 0
       @ spd = $spd + 1
       endif
cd ..
end
wait
cd RESULTS1
cp ../*.f .
rm -r -f fouout*
ifort extfor.f
mv a.out forexe
ifort foucom.f
mv a.out fouexe
 set r = 1
 while($r <= 6)
   echo "for rad station = $r"
   set s = 50
   foreach i (0)
     @ s = $s + 5
     set b = 250
     foreach m (0 1 2 3)
         @ b =  $b - 25
```

```
        cp force$i$m fort.1
# Call FORTRAN module to extract force data
./forexe <<EOF
$r
EOF
         cp fort.13 fourier.inp
#Call FORTRAN module to compute Fourier Coef
./fouexe <<EOF
$b $s
EOF
         echo "$i$m done"
cp fourier.out fouout$i$m$r
         end
    end
wait
@ r = $r + 1
End
```